Homework 5: Models and Model Comparisons

Max Campbell

Task 1: Conceptual Questions

- What is the purpose of using cross-validation when fitting a random forest model?

Generally, when fitting a random forest model, we want to randomly subset predictors to make sure that no predictor overwhelms the model and significantly effects the output. As such, we need to choose how many predictors are subset each time. This is where cross-validation comes into play. By implementing cross-validation and obtaining the most optimized tuning parameter we can improve our predictions.

- Describe the bagged tree algorithm.

In the bagged tree algorithm, we begin by bootstrapping some number of samples B. From there, we can fit a tree to each sample to have B number of trees. Then, from there, we can find response for each tree and then combine them to aggregate a final prediction. In practice, this combination is typically the average of all the sample predictions.

- What is meant by general linear model?

A general linear model is the family of models that we use more conventionally. For example, SLRs, MLRs, ANOVA and ANCOVA models all fall under this family (so long as all effects are fixed and not random). The main characteristic of these models is assuming that the response variable follows a normal distribution.

- When fitting a multiple linear regression model, what does adding an interaction term do? That is, what does it allow the model to do differently as compared to when it is not included in the model?

The main reason to include an interaction variable is to allow the effect of one variable to be dependent on another variable. For example, in a plant growth study, the amount of sunlight and temperature may be heavily related to one another, so an interaction term may be beneficial to include in the model.

- Why do we split our data into a training and test set?

Splitting the model into a training set and test set allows us to see how well the model may perform if it was tasked with prediction future observations. By having test data, we can compare a model's predictions to the actual outcome to measure performance.

Task 2: Data Prep

Packages and Data

```
library(tidyverse)
library(tidymodels)
library(caret)
library(yardstick)
library(glmnet)

heart <- as_tibble(read.csv("heart.csv", header = TRUE))
```

Question 1

```
summary(heart)
```

```
      Age            Sex            ChestPainType        RestingBP     
 Min.   :28.00   Length:918         Length:918         Min.   :  0.0  
 1st Qu.:47.00   Class :character   Class :character   1st Qu.:120.0  
 Median :54.00   Mode  :character   Mode  :character   Median :130.0  
 Mean   :53.51                                         Mean   :132.4  
 3rd Qu.:60.00                                         3rd Qu.:140.0  
 Max.   :77.00                                         Max.   :200.0  
  Cholesterol      FastingBS       RestingECG            MaxHR      
 Min.   :  0.0   Min.   :0.0000   Length:918         Min.   : 60.0  
 1st Qu.:173.2   1st Qu.:0.0000   Class :character   1st Qu.:120.0  
 Median :223.0   Median :0.0000   Mode  :character   Median :138.0  
 Mean   :198.8   Mean   :0.2331                      Mean   :136.8  
 3rd Qu.:267.0   3rd Qu.:0.0000                      3rd Qu.:156.0  
 Max.   :603.0   Max.   :1.0000                      Max.   :202.0  
 ExerciseAngina        Oldpeak          ST_Slope          HeartDisease    
 Length:918         Min.   :-2.6000   Length:918         Min.   :0.0000  
 Class :character   1st Qu.: 0.0000   Class :character   1st Qu.:0.0000  
 Mode  :character   Median : 0.6000   Mode  :character   Median :1.0000  
```

```
Mean   : 0.8874                    Mean   :0.5534
3rd Qu.: 1.5000                    3rd Qu.:1.0000
Max.   : 6.2000                    Max.   :1.0000
```

The `HeartDisease` variable is quantitative. This does not make sense, since an individual either has a heart disease or they don't, so it is more intuitive to consider this a categorical variable.

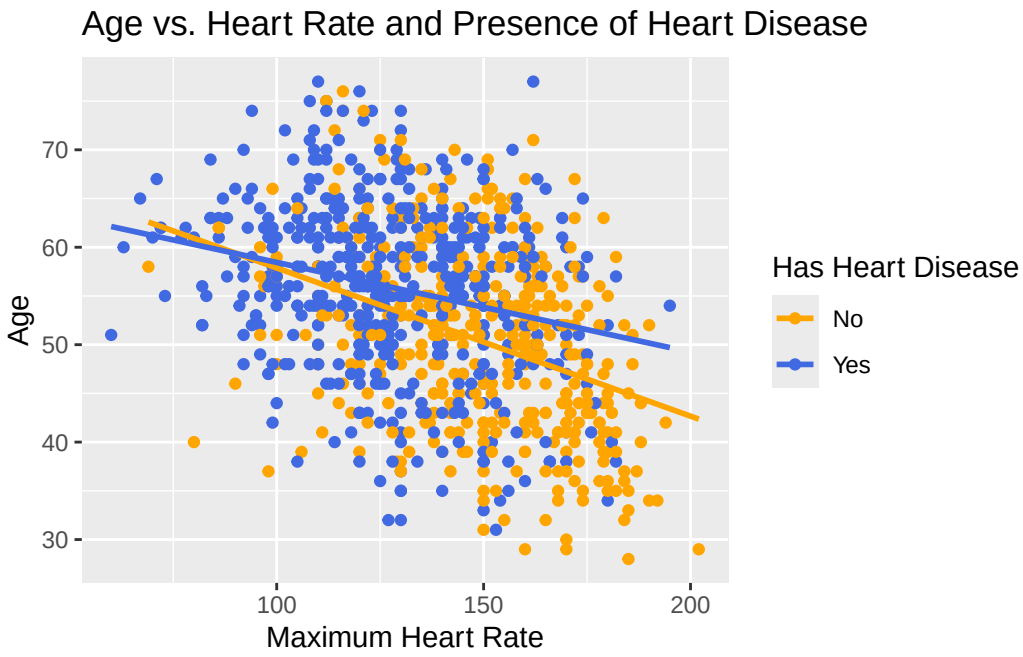
Question 2

```
#Subset dataset into relevant data of the correct type

heart_new <- heart |>
  mutate(HasHeartDisease = as.factor(HeartDisease)) |>
  select(-c(ST_Slope, HeartDisease))

#Change names of factor levels in HasHeartDisease to improve plot
# outputs later down the line

levels(heart_new$HasHeartDisease) <- c("No", "Yes")
```

Task 3: EDA

Question 1

```
#Plot Age vs. Heart Rate, grouped by Heart Disease status

ggplot(data = heart_new, aes(x = MaxHR, y = Age, color = HasHeartDisease)) +
  geom_point() +
  geom_smooth(method = lm, se = FALSE, formula = y ~ x) +
  labs(x = "Maximum Heart Rate", y = "Age",
       title = "Age vs. Heart Rate and Presence of Heart Disease",
       color = "Has Heart Disease") +
  scale_color_manual(values = c("orange", "royalblue"))
```



Question 2

Based on our output above, it looks like the presence of heart disease makes a sizable impact on the regression line. As such, an interaction term appears necessary as heart disease may have a notable effect on an individuals maximum heart rate.

Task 4: Testing and Training

```
#Set random seed so results are reproducible
set.seed(101)

#Split data into a testing set and training set
heart_split <- initial_split(heart_new, prop = 0.8)
test <- testing(heart_split)
train <- training(heart_split)
```

Task 5: OLS and LASSO

Question 1

```
#Fit an ordinary least squares multiple linear regression model
ols_mlr <- lm(Age ~ MaxHR + HasHeartDisease + MaxHR:HasHeartDisease, data = train)
summary(ols_mlr)
```

```
Call:
lm(formula = Age ~ MaxHR + HasHeartDisease + MaxHR:HasHeartDisease,
    data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-22.7703  -5.7966   0.4516   5.7772  20.6378

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              75.58896    3.07510  24.581  < 2e-16 ***
MaxHR                    -0.16992    0.02064  -8.233 8.43e-16 ***
HasHeartDiseaseYes       -8.58502    3.83433  -2.239  0.02546 *
MaxHR:HasHeartDiseaseYes  0.08343    0.02716   3.072  0.00221 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.478 on 730 degrees of freedom
Multiple R-squared:  0.1839,    Adjusted R-squared:  0.1806
F-statistic: 54.84 on 3 and 730 DF,  p-value: < 2.2e-16
```

Question 2

```
# Using tidymodels functionality, calculate the RMSE
ols_mlr_rmse <- yardstick::rmse_vec(test$Age, predict(ols_mlr, newdata = test))
ols_mlr_rmse
```

```
[1] 9.100206
```

Question 3

```r
#Create a 10-fold CV
train_folds <- vfold_cv(train, 10)

#Create an interaction model using a LASSO parameter
LASSO_recipe <- recipe(Age ~ MaxHR + HasHeartDisease, data = train) |>
  step_dummy(HasHeartDisease) |>
  step_normalize(MaxHR) |>
  step_interact(~MaxHR:starts_with("HasHeartDisease_"))

#Display recipe properties
LASSO_recipe
```

```
-- Recipe ----------------------------------------------------------------------
```

```
-- Inputs

Number of variables by role

outcome:   1
predictor: 2
```

```
-- Operations

* Dummy variables from: HasHeartDisease

* Centering and scaling for: MaxHR

* Interactions with: MaxHR:starts_with("HasHeartDisease_")
```

Question 4

```r
#Set spec and workflow
LASSO_spec <- linear_reg(penalty = tune(), mixture = 1) |>
  set_engine("glmnet")

LASSO_workflow <- workflow() |>
  add_recipe(LASSO_recipe) |>
  add_model(LASSO_spec)

#Set grid
LASSO_grid <- LASSO_workflow |> #Using 200 as it seems to be a reasonable value
  tune_grid(resamples = train_folds, grid = grid_regular(penalty(), levels = 200))

#Get the model with the best RMSE and fit it to our training data
rmse_best_fit <- LASSO_grid |>
  select_best(metric = "rmse")

LASSO <- LASSO_workflow |>
  finalize_workflow(rmse_best_fit) |>
  fit(train)

#Report results
tidy(LASSO)
```

```
# A tibble: 4 x 3
  term                          estimate penalty
  <chr>                            <dbl>   <dbl>
1 (Intercept)                      52.5   0.0174
2 MaxHR                            -4.22  0.0174
3 HasHeartDisease_Yes               2.75  0.0174
4 MaxHR_x_HasHeartDisease_Yes       2.00  0.0174
```

Question 5

The model coefficients in the LASSO model are very different from the MLR model, so we might expect the RMSE to be different since the observations did not change.

Question 6

```
#Ordinary Least Squares RMSE
ols_mlr_rmse
```

```
[1] 9.100206
```

```
#LASSO RMSE
LASSO |>
  predict(test) |>
  pull() |>
  rmse_vec(truth = test$Age)
```

```
[1] 9.091133
```

The results are very similar! The LASSO model has the slightest edge in this run, but both models are performing essentially the same.

Question 7

We standardized the values for our LASSO model, but did not do so for the MLR model, which is likely the biggest factor in why our coefficients are so different. As such, the line of best fit likely falls within a similar spot relative to the observations in both datasets, and thus our RMSE would be similar as well.

Task 6: Logistic Regression

Question 1

I am proposing two different logistic regression models for this dataset. The first model will attempt to use a person's age and their maximum heart rate as predictors. The second model will use those same predictors with the addition of the resting Blood Pressure, Cholesterol, and Sex predictors. The purpose of comparing these two models is to allow us to have insight on how significantly our model relies on Age and maximum Heart Rate as predictors. We might expect to see a large difference in accuracy if one of our added predictors in the second model appears to be stronger.

Now, let's begin:

```r
#Using our training folds from the previous exercise,
# we will fit two logistic models via a 10-fold CV
#Define recipes
log_reg1 <- recipe(HasHeartDisease ~ Age + MaxHR, data = train) |>
  step_normalize(Age, MaxHR)

log_reg2 <- recipe(HasHeartDisease ~ Age + MaxHR + RestingBP + Cholesterol + Sex,
                   data = train) |>
  step_normalize(Age, MaxHR, RestingBP, Cholesterol) |>
  step_dummy(Sex)

#Set spec for both models
logistic_spec <- logistic_reg() |>
  set_engine("glm")

#Create workflows for each model
workflow1 <- workflow() |>
  add_recipe(log_reg1) |>
  add_model(logistic_spec)

workflow2 <- workflow() |>
  add_recipe(log_reg2) |>
  add_model(logistic_spec)

#Fit CV folds
fit1 <- workflow1 |>
  fit_resamples(train_folds, metrics = metric_set(accuracy, mn_log_loss))

fit2 <- workflow2 |>
  fit_resamples(train_folds, metrics = metric_set(accuracy, mn_log_loss))

#Collect metrics and identify best performing model
rbind(fit1 |>
        collect_metrics(),
      fit2 |>
        collect_metrics()
      ) |>
  mutate(Model = c("fit1", "fit1", "fit2", "fit2")) |>
  select(Model, everything())
```

```
# A tibble: 4 x 7
  Model .metric     .estimator  mean     n std_err .config
```

```
  <chr> <chr>       <chr>      <dbl> <int>   <dbl> <chr>
1 fit1  accuracy    binary     0.688    10  0.0180 Preprocessor1_Model1
2 fit1  mn_log_loss binary     0.596    10  0.0121 Preprocessor1_Model1
3 fit2  accuracy    binary     0.715    10  0.0125 Preprocessor1_Model1
4 fit2  mn_log_loss binary     0.561    10  0.0205 Preprocessor1_Model1
```

It does appear that our model with more predictors is a better fit for this dataset. This isn't surprising. However, it is only about 3% more accurate than the model with only two predictors! This tells us that a person's age and maximum heart rate appear to be strong predictors for heart disease, while a person's sex, resting blood pressure, and cholesterol levels can help to paint a more detailed picture in cases where the model is unsure what to predict (as in, produces a value near 0.5).

Question 2

```
#Use our test data to see how well our model performs!
#Fit to training data
train_fit <- workflow2 |>
  fit(train)

#Show metrics with the output
workflow2 |>
  last_fit(heart_split, metrics = metric_set(accuracy, mn_log_loss)) |>
  collect_metrics()
```

```
# A tibble: 2 x 4
  .metric     .estimator .estimate .config
  <chr>       <chr>          <dbl> <chr>
1 accuracy    binary         0.739 Preprocessor1_Model1
2 mn_log_loss binary         0.532 Preprocessor1_Model1
```

```
conf_mat(test |>
                  mutate(estimate = train_fit |>
                           predict(test) |>
                           pull()), #data
                HasHeartDisease, #truth
                estimate #fitted estimates
                )
```

```
          Truth
Prediction No Yes
       No  66  20
       Yes 28  70
```

It appears our model gets the prediction correct about 73.9% of the time! That's not a bad start.

Question 3

The sensitivity of our model is the rate at which our model correctly predicted that the individual had heart disease, given that the truth is that they had heart disease. In our model, the sensitivity is calculated as $70/(20 + 70) = 0.777$. This means that our model correctly predicted that an individual with heart disease had heart disease 77.7% of the time.

The specificity of our model is the rate at which our model correctly predicted that the individual did not have heart disease, given that the truth is that they did not have heart disease. In our model, the specificity is calculated as $66/(66 + 28) = 0.702$. This means that our model correctly predicted that an individual without heart disease did not have heart disease 70.2% of the time.